METHOD, SYSTEM, AND PROGRAM FOR HANDLING ANCHOR TEXT

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0001] The present invention is related to handling anchor text for information retrieval.

2. Description of the Related Art

[0002] The World Wide Web (also known as WWW or the "Web") is a collection of some Internet servers that support Web pages that may include links to other Web pages. A Uniform Resource Locator (URL) indicates a location of a Web page. Also, each Web page may contain, for example, text, graphics, audio, and/or video content. For example, a first Web page may contain a link to a second Web page. When the link is selected in the first Web page, the second Web page is typically displayed.

[0003] A Web browser is a software application that is used to locate and display Web pages. Currently, there are billions of Web pages on the Web.

[0004] Web search engines are used to retrieve Web pages on the Web based on some criteria (e.g., entered via the Web browser). That is, Web search engines are designed to return relevant Web pages given a keyword query. For example, the query "HR" issued against a company intranet search engine is expected to return relevant pages in the intranet that are related to Human Resources (HR). The Web search engine uses indexing techniques that relate search terms (e.g., keywords) to Web pages.

[0005] An anchor may be described as a link or path to a document (e.g., a URL). Anchor text may be described as text associated with a path or link (e.g., a URL) that points to a document. For example, anchor text may be text that labels or encloses hypertext text links in Web documents. Anchor text is collected by Web search engines and is associated with target documents. Also, the anchor text and target documents are indexed together.

[0006] Web search engines use context information (e.g., title, summary, language, etc.) to enrich search results. This provides a user with screened search results. Anchor text, however, may not be relevant for use as context information. For example, anchor text may be in a different language than the target document, and use of the anchor text without further processing may result in, for example, a Japanese title for an English document. Moreover, anchor text may not be related to the content of the document. For instance, anchor text may contain common words (e.g., "Click here") that occur often and are used primarily for navigation, but which do not have any meaningful value as a title. Also, anchor text may be inaccurate, impolite or may contain slang, (e.g., an anchor to a "Network Security Guide" has anchor text "Looking for Trouble?").

[0007] Moreover, generation of context information is especially difficult when the contents of a Web page can not be retrieved (e.g., due to server outage, incompleteness of the retrieval of Web pages for processing by the search engine, robots.txt prohibiting access) or when a document is retrieved but cannot be analyzed (e.g., because the file is a video/audio/multimedia file, is in an unknown or unsupported format, is ill-formed or is password protected).

[0008] Most search engines display only a Uniform Resource Locator (URL) in the absence of content of a Web page. That, however, makes it hard for the user to capture the usefulness of a search result without looking at the Web page itself.

[0009] Thus, there is a need for improved document processing to provide context information for documents, such as Web pages.

SUMMARY OF THE INVENTION

[0010] Provided are a method, system, and program for processing anchor text. A set of anchors that point to a target document is formed. Anchors with same anchor text are grouped together. Information is computed for each group. Context information is generated for the target document based on the computed information.

BRIEF DESCRIPTION OF THE DRAWINGS

Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

FIG. 1 illustrates, in a block diagram, a computing environment in accordance with certain implementations of the invention.

FIG. 2 illustrates logic implemented to prepare anchors for processing in accordance with certain implementations of the invention.

FIGs. 3A and 3B illustrate logic implemented to process anchor text in accordance with certain implementations of the invention.

FIG. 4 illustrates logic for performing a document search in accordance with certain implementations of the invention.

FIG. 5 illustrates an architecture of a computer system that may be used in accordance with certain implementations of the invention.

DETAILED DESCRIPTION

[0011] In the following description, reference is made to the accompanying drawings which form a part hereof and which illustrate several implementations of the present invention. It is understood that other implementations may be utilized and structural and operational changes may be made without departing from the scope of the present invention.

[0012] Certain implementations of the invention make a document available for searching by indexing anchor text instead of or in addition to content. Certain implementations generate context information from the anchor text of anchors that point to a document. For instance, at least a portion of anchor text may be designated as a title or a summary of the document. However, it may be difficult to identify meaningful anchor text because anchor text may be in a different language than the target document, anchor text may not be related to the content of the document, or anchor text may be inaccurate, impolite or may contain slang. Furthermore, special care is taken to remove noise in the anchor text that does not have meaningful value as, for example, a title (e.g., URLs, file names, navigational text such as "next column").

[0013] Thus, certain implementations of the invention process raw anchor text to obtain high quality titles and summaries. Certain implementations of the invention distill raw anchor text to obtain high quality data that can be used for generating title or summary data for search result items. The result of the raw anchor text processing improves the overall search quality and, therefore, user experience in a document retrieval system.

[0014] FIG. 1 illustrates, in a block diagram, a computing environment in accordance with certain implementations of the invention. A client computer 100 is connected via a network 190 to a server computer 120. The client computer 100 may comprise any computing device known in the art, such as a server, mainframe, workstation, personal computer, hand held computer, laptop telephony device, network appliance, etc. The network 190 may comprise any type of network, such as, for example, a Storage Area Network (SAN), a Local Area Network (LAN), Wide Area Network (WAN), the Internet, an Intranet, etc. The client computer 100 includes system memory 104, which may be implemented in volatile and/or non-volatile devices. One or more client applications 110 and a viewer application 112 may execute in the system memory 104. The viewer application 112 provides an interface that enables searching of a set of documents (e.g., stored in one or more data stores 170. In certain implementations, the viewer application 112 is a Web browser.

[0015] The server computer 120 includes system memory 122, which may be implemented in volatile and/or non-volatile devices. A search engine 130 executes in the system memory 122. In certain implementations, the search engine includes a crawler component 132, a static rank component 134, a document analysis component 136, a duplicate detection component 138, an anchor text component 140, and an indexing component 142. The anchor text component 140 includes a context information generator 141. Although components 132, 134, 136, 138, 140, 141, and 142 are illustrated as separate components, the functionality of components 132, 134, 136, 138, 140, 141, and 142 may be implemented in fewer or more or different components than illustrated. Additionally, the functionality of the components 132, 134, 136, 138, 140, 141, and 142 may be implemented at a Web application server computer or other server

computer that is connected to the server computer 120. Additionally, one or more server applications 160 execute in system memory 122.

[0016] The server computer 120 provides the client computer 100 with access to data in at least one data store 170 (e.g., a database). Although a single data store 170 is illustrated, for ease of understanding, data in data store 170 may be stored in data stores at other computers connected to server computer 120.

[0017] Also, an operator console 180 executes one or more applications 182 and is used to access the server computer 120 and the data store 170.

[0018] The data store 170 may comprise an array of storage devices, such as Direct Access Storage Devices (DASDs), Just a Bunch of Disks (JBOD), Redundant Array of Independent Disks (RAID), virtualization device, etc. The data store 170 includes data that is used with certain implementations of the invention.

[0019] FIG. 2 illustrates logic implemented to prepare anchors for processing in accordance with certain implementations of the invention. Control begins at block 200 with anchor text being associated with each anchor. This may be done, for example, by each user who creates an anchor. An anchor may be described as a path or link (e.g., a URL) from a source document to a target document.

[0020] In block 202, documents that are to be indexed by the search engine 130 are obtained. In certain implementations, the documents are published or pushed (e.g., as may be the case with newspaper articles) to the indexing component 142. In certain implementations, the crawler component 132 discovers, fetches, and stores the documents. In certain implementations, the crawler component 132 may discover documents based on, for example, certain criteria (e.g., documents were accessed within the last month). Additionally, the crawler component 132 may discover documents in one or more data stores connected directly (e.g., data store 170) or indirectly (e.g., connected to server computer 120 via another computing device (not shown)) to server computer 120. In certain implementations, the crawler component 132 discovers, fetches, and stores Web pages in data store 170. These stored documents may be referred to as a "collection of documents".

[0021] In block 204, the document analysis component 136 performs per document analysis. In particular, the document analysis component 136 reviews the stored documents, parses and tokenizes the documents, and determines, for each document, the language in which each document was written, extracts anchor text, and performs other tasks such as, categorization and classification of the documents. The language information is stored for later use. For example, the document analysis component 136 determines whether the primary language used in the document is English, Japanese, German, etc. As part of extracting the anchor text, the document analysis component 136 also associates a proximity class with each anchor. A proximity class may be described as specifying how close a source document is to a target document (e.g., whether they are on the same server, and if so, whether they are in the same directory). Also, the extracted anchor text is ready to be processed by the anchor text component 140.

[0022] In block 206, the static rank component 134 reviews the stored documents and assigns a rank to the documents. The rank may be described as the importance of the source document relative to other documents that have been stored by the crawler component 132. Any type of ranking technique may be used. For example, documents that are accessed more frequently may receive a higher rank.

[0023] In block 208, the context information generator 141 sorts the anchors by target document. This results in the set of anchors for a target document being grouped together in a set for further processing. Each set is separately processed for each target document, as will be described with reference to FIGs. 3A and 3B.

[0024] FIGs. 3A and 3B illustrate logic implemented to process anchor text in accordance with certain implementations of the invention. Control begins at block 300 with the context information generator 141 determining a predominant language of the source documents of the anchors pointing to the target document in the set of anchors for the target document. In certain implementations, if more than a configurable percentage of source documents have the same language, anchors from the set whose source document language is different from the predominant language are removed. A

configurable percentage may be described as a percentage that may be modified by, for example, a system administrator or another application program.

[0025] In block 302, the context information generator 141 removes anchors that have anchor text that contains a path (e.g., URL) or a portion of a path to the target document. In block 304, the context information generator 141 removes anchor text based on whether and in which order or combination the anchor text contains words from a configurable set of words (e.g., anchor text that contains only words from the configurable set, that contains at least a number of words from the configurable set or contains words from the configurable set in a certain order may be removed). The configurable set of words may be, for example, determined by a system administrator. For example, the configurable set of words may include stop words, such as "click here" or "the".

[0026] In block 306, the context information generator 141 sorts the set of anchors by anchor text and groups together anchors having the same anchor text. In block 308, the context information generator 141 computes a weighted sum of occurrences of the text for each group. The weight of each individual occurrence of the text may be determined by the proximity class of the anchor. For example, if a first document has a proximity class A, a second document has a proximity class B, and a third document has proximity class C, and classes A, B and C have weight 10, 5, and 2 respectively, the weighted sum is 17.

[0027] In block 310, the context information generator 141 computes an accumulated rank for each group. That is, each anchor in the group contributes to this rank according to the rank of its source document and its proximity class. For example, if a first document has a proximity class A, a second document has a proximity class B, and a third document has proximity class C, and classes A, B and C have weight 10, 5, and 2 respectively, if the first, second and third documents have static ranks 9,13,and 16, respectively, and if the accumulated rank is computed by weighted average, the accumulated rank is (9*10 + 13*5 + 16*2) / (10 + 5 + 2) = 187 / 17 = 11. Other

techniques of computing accumulated rank include minimum, maximum, or both of these in combination with preferring the ranks of one proximity class over the others, etc.

[0028] In block 312, the context information generator 141 computes a linguistic score for each group. In certain implementations, this score may be computed by a linguistic analysis of the text that scores the text for displayability as a title. For example, displayability as a title may be determined by considering the number of words in the text (e.g, a title should be brief), further linguistic analysis of the text, statistical analysis of each word or the number of occurrences of the words in all anchors in the set of anchors that point to a target document or the similarity of the anchor to the content of the target document when the target document is available for access.

[0029] In block 314, the context information generator 141 computes a combined relevance score from the weighted sum of occurrences, the accumulated static rank, and the linguistic score for each group.

[0030] In block 316, the context information generator 141 generates context information for the target document. In certain implementations, the context information generator 141 selects the text of the group with the highest combined relevance score as a pseudo-title, composes an anchor-based static summary for the target document from the anchor texts of the n groups with the highest relevance score, and concludes the language of T from the predominant source language.

[0031] Once anchor text processing is completed, the indexing component 142 generates an index, using the processed anchor text.

[0032] FIG. 4 illustrates logic for performing a document search in accordance with certain implementations of the invention. Control begins at block 400 with a user submitting a search request via the viewer application 112. In block 402, the search engine 130 executes the search request. In block 404, the search engine returns search results that include the anchor text processing and other processing described in FIGs. 2A and 2B. In block 406, the viewer application 112 displays the search results.

[0033] Thus, certain implementations of the invention provide a technique for generating high-quality context information for search result items from a collection of anchors. In

certain implementations, an analysis of each document is performed to identify the language in which the document was written, a global analysis of all documents is performed to assign a static rank to each document, and anchors are sorted by target document to obtain for each target document a logical collection of all anchors that point to the target document. For each collection of anchors pointing to a target document, the following may be performed: analysis of the distribution of the languages of the source documents; pruning of anchors from the collection based on the language distribution; noise filtering based on stop word and URL detection; classification of each anchor according to the proximity of the source to the target; and, assigning of a weight to each proximity class. Additionally, each anchor may be scored based on linguistic analysis of anchor text of the anchor. Furthermore, relevance-ordering of remaining unique anchor texts (i.e., the same text can be on different anchors) may be performed based on the weighted sum of occurrences in each proximity class, the accumulated rank of all source documents, and the linguistic score of the text.

[0034] The results of the anchor text processing are high quality titles, summaries, and other contextual information (e.g, the most likely language for each target). For search results where the target document is not available, this context information may be displayed to the user. If the target document itself is available, the generated context information may be used to enrich the information gained from the target document (e.g., by finding similarities between the document and its anchors).

Additional Implementation Details

[0035] The described techniques for handling anchor text may be implemented as a method, apparatus or article of manufacture using standard programming and/or engineering techniques to produce software, firmware, hardware, or any combination thereof. The term "article of manufacture" as used herein refers to code or logic implemented in hardware logic (e.g., an integrated circuit chip, Programmable Gate Array (PGA), Application Specific Integrated Circuit (ASIC), etc.) or a computer readable medium, such as magnetic storage medium (e.g., hard disk drives, floppy disks,,

tape, etc.), optical storage (CD-ROMs, optical disks, etc.), volatile and non-volatile memory devices (e.g., EEPROMs, ROMs, PROMs, RAMs, DRAMs, SRAMs, firmware, programmable logic, etc.). Code in the computer readable medium is accessed and executed by a processor. The code in which various implementations are implemented may further be accessible through a transmission media or from a file server over a network. In such cases, the article of manufacture in which the code is implemented may comprise a transmission media, such as a network transmission line, wireless transmission media, signals propagating through space, radio waves, infrared signals, etc. Thus, the "article of manufacture" may comprise the medium in which the code is embodied. Additionally, the "article of manufacture" may comprise a combination of hardware and software components in which the code is embodied, processed, and executed. Of course, those skilled in the art will recognize that many modifications may be made to this configuration without departing from the scope of the present invention, and that the article of manufacture may comprise any information bearing medium known in the art.

[0036] The logic of FIGs. 2, 3A, 3B, and 4 describes specific operations occurring in a particular order. In alternative implementations, certain of the logic operations may be performed in a different order, modified or removed. Moreover, operations may be added to the above described logic and still conform to the described implementations. Further, operations described herein may occur sequentially or certain operations may be processed in parallel, or operations described as performed by a single process may be performed by distributed processes.

[0037] The illustrated logic of FIGs. 2, 3A, 3B, and 4 may be implemented in software, hardware, programmable and non-programmable gate array logic or in some combination of hardware, software, or gate array logic.

[0038] FIG. 5 illustrates an architecture of a computer system that may be used in accordance with certain implementations of the invention. For example, client computer 100, server computer 120, and/or operator console 180 may implement computer architecture 500. The computer architecture 500 may implement a processor 502 (e.g., a

microprocessor), a memory 504 (e.g., a volatile memory device), and storage 510 (e.g., a non-volatile storage area, such as magnetic disk drives, optical disk drives, a tape drive, etc.). An operating system 505 may execute in memory 504. The storage 510 may comprise an internal storage device or an attached or network accessible storage. Computer programs 506 in storage 510 may be loaded into the memory 504 and executed by the processor 502 in a manner known in the art. The architecture further includes a network card 508 to enable communication with a network. An input device 512 is used to provide user input to the processor 502, and may include a keyboard, mouse, pen-stylus, microphone, touch sensitive display screen, or any other activation or input mechanism known in the art. An output device 514 is capable of rendering information from the processor 502, or other component, such as a display monitor, printer, storage, etc. The computer architecture 500 of the computer systems may include fewer components than illustrated, additional components not illustrated herein, or some combination of the components illustrated and additional components.

[0039] The computer architecture 500 may comprise any computing device known in the art, such as a mainframe, server, personal computer, workstation, laptop, handheld computer, telephony device, network appliance, virtualization device, storage controller, etc. Any processor 502 and operating system 505 known in the art may be used.

[0040] The foregoing description of implementations of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto. The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many implementations of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.